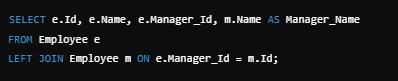
**HashedIn by Deloitte Data Engineer Interview Guide – Experienced 3+**

**Technical Round 1**

**1. Add a new column with manager names for each employee**

Given a table structure with Id, Name, and Manager\_Id, use a **self-join** to find the manager’s name.

**SQL Query**:



**2. Identify who is a manager and who is not**

To find managers (who have employees reporting to them) and non-managers:

**SQL Query**:

SELECT DISTINCT Manager\_Id AS Manager FROM Employee WHERE Manager\_Id IS NOT NULL;

SELECT Id FROM Employee WHERE Id NOT IN (SELECT DISTINCT Manager\_Id FROM Employee WHERE Manager\_Id IS NOT NULL);

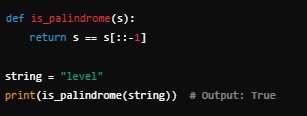
**3. Add a new column with the average salary by department**

**SQL Query**:

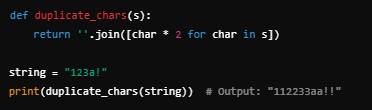


**4. Check if a string is a palindrome**

**Python Code**:

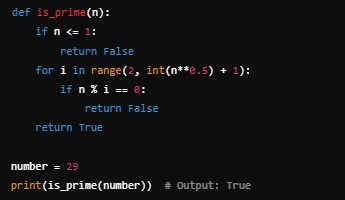


**5. Duplicate characters in a string ("123a!" → "112233aa!!") Python Code**:



**6. Check if a number is prime**

**Python Code**:



**Technical Round 2**

**1. Processing 1 TB of data in Spark**

Steps to efficiently process 1 TB of data:

 Use partitioning and bucketing to distribute data evenly.

 Leverage columnar file formats like Parquet or ORC.

 Optimize shuffle operations with coalesce and reduceByKey instead of groupByKey.

 Utilize broadcast joins for small lookup tables.

 Enable Adaptive Query Execution (AQE) to dynamically optimize the query plan.

 Use caching only for reused datasets.

 Tune executor memory and number of cores based on the cluster size.

**2. Designing a Data Warehouse (DWH) Problem Statement**

Consider designing a DWH for an e-commerce platform:

 **Fact Tables**:

 Sales\_Fact: Stores transaction details (product\_id, customer\_id, quantity, revenue, date).

 Inventory\_Fact: Tracks inventory levels by warehouse.

 **Dimension Tables**:

 Product\_Dim: product\_id, product\_name, category.

 Customer\_Dim: customer\_id, name, address.

 Date\_Dim: date\_key, day, month, year, quarter.

 **Schema Design**:

 Use a star schema for easy navigation.

 Apply partitioning on large tables based on date for efficient querying.

 Ensure surrogate keys for dimensions.

**3. How would you design a data pipeline to handle semi-structured and unstructured data?**

 Explain the design using distributed file systems (HDFS, S3), data processing frameworks (Spark, Flink), and NoSQL databases (HBase, MongoDB).

 Include stages: Ingestion (Kafka, Kinesis), Data Lake (Parquet/ORC storage), Transformation (Spark DataFrame, PySpark), and Serving Layer (Redshift, Snowflake, Elasticsearch).

 Discuss schema inference and schema-on-read strategies.

**4. Explain the differences between Spark's shuffle and broadcast join. When would you use each?**

 **Shuffle Join**:

 Used when both datasets are large.

 Requires shuffling of data across nodes.

 Slower and resource-intensive.

 **Broadcast Join**:

 Broadcasts a smaller dataset to each executor.

 Used when one dataset is small enough to fit in memory.

 Reduces shuffle and improves performance.

**When to use:**

 Prefer broadcast joins when a small lookup table is involved.

 Use shuffle join for larger datasets, optimizing with partitioning strategies.

**5. What strategies can you use to handle skewed data in Spark?**

 Salting: Add a random key to distribute data evenly across partitions.

 Increase parallelism: Adjust the number of partitions using repartition().

 Broadcast smaller tables: Use broadcast joins.

 Adaptive Query Execution (AQE): Enable Spark’s dynamic partitioning feature.

**6. Explain Adaptive Query Execution (AQE) in Spark 3.x.**

 AQE optimizes queries dynamically at runtime based on the actual data processed.

 **Key Features**:

Dynamic partition pruning: Reduces partitions scanned based on filter conditions.

 Join optimization: Automatically switches join strategies (broadcast, shuffle)

during execution.

 Coalescing shuffle partitions: Reduces the number of shuffle partitions.

**7. How would you optimize a Spark job that takes too long to run in production?**

 **Profiling tools**: Use Spark UI to identify bottlenecks.

 **Techniques**:

o Cache frequently used DataFrames.

o Use columnar file formats like Parquet/ORC with predicate pushdown.

o Avoid using collect() or count() on large datasets.

O Reduce shuffle operations with efficient transformations (reduceByKey over groupByKey).

**8. Explain how you would implement a Slowly Changing Dimension (SCD) Type 2 in**

**Spark.**

 Use merge operations with Delta Lake or Apache Hudi.

 Steps:

 Load current data as the target DataFrame.

 Load incoming data as the source DataFrame.

 Identify new records, unchanged records, and updated records.

 Assign start\_date, end\_date, and current flag to version data.

**9. How do you monitor and debug Spark applications in production?**

 **Spark UI**: Provides DAG visualization and job stages.

 **Ganglia/Prometheus**: Used for resource monitoring.

 **Event logs**: Enable detailed job logging.

 **Metrics**: Monitor executor memory usage, garbage collection, and shuffle read/write.

**10. How do you design a scalable and fault-tolerant data warehouse on a cloud Platform?**

 **Storage**: Use S3 or Azure Blob Storage for data lakes.

 **Compute**: Spark on EMR or Databricks for processing.

 **Metadata Management**: Hive Metastore or AWS Glue.

 **Query Engine**: Presto, Redshift, or Synapse Analytics.

 **Fault Tolerance**: Leverage **checkpointing** and **retry policies**.